

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

50023-135
June 4, 2001
MIYAKE, ET AL.
McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application:

2000年12月11日

出 願 番 号
Application Number:

特願2000-375641

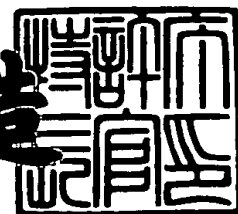
出 願 人
Applicant(s):

松下電器産業株式会社

2001年 3月30日

特許庁長官
Commissioner,
Patent Office

及 川 耕 造



出証番号 出証特2001-3025547

【書類名】 特許願

【整理番号】 2022520408

【提出日】 平成12年12月11日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 11/28

【発明者】

【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

【氏名】 三宅 秀明

【発明者】

【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

【氏名】 枯木 正吉

【発明者】

【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

【氏名】 鶴本 克己

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100097445

【弁理士】

【氏名又は名称】 岩橋 文雄

【選任した代理人】

【識別番号】 100103355

【弁理士】

【氏名又は名称】 坂口 智康

【選任した代理人】

【識別番号】 100109667

【弁理士】

【氏名又は名称】 内藤 浩樹

【先の出願に基づく優先権主張】

【出願番号】 特願2000- 94064

【出願日】 平成12年 3月30日

【手数料の表示】

【予納台帳番号】 011305

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9809938

【書類名】 明細書

【発明の名称】 デバッグ支援装置、デバッグ支援方法及びそのプログラムを記録したコンピュータ読み取り可能な記録媒体

【特許請求の範囲】

【請求項 1】 汎用 OS 上でリアルタイム OS をシミュレートする OS シミュレータがリンクされたアプリケーションと、

前記 OS シミュレータで変更され、前記アプリケーションのタスク実行を制御する OS 管理情報を記憶する記憶手段と、

前記 OS シミュレータで変更される OS 管理情報を参照または変更する OS デバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記アプリケーションと前記 OS デバッガとで共有され、前記記憶手段に記憶された OS 管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、

前記記憶手段の OS 管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、

前記書込手段で OS 管理情報が書き込まれた共有管理情報を読み出す読出手段とを備えることを特徴とするデバッグ支援装置。

【請求項 2】 前記書込手段は、前記アプリケーション側に備えられ、前記 OS シミュレータの要求に応じて前記 OS 管理情報の変更を前記共有管理情報に書き込み、

前記読出手段は、前記 OS デバッガ側に備えられ、当該 OS デバッガの要求に応じて前記書込手段で OS 管理情報が書き込まれた共有管理情報を読み出すことを特徴とする請求項 1 に記載のデバッグ支援装置。

【請求項 3】 前記読出手段は、任意のタイミングで前記共有ファイルから共有管理情報を読み出すことを特徴とする請求項 2 に記載のデバッグ支援装置。

【請求項 4】 前記読出手段は、所定の周期で前記共有ファイルから共有管理情報を読み出すことを特徴とする請求項 2 に記載のデバッグ支援装置。

【請求項 5】 汎用 OS 上でリアルタイム OS をシミュレートする OS シミ

ュレータがリンクされたアプリケーションと、

前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルと、

外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込手段と、

前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込手段で書き込まれたデバッグ命令を読み出すデバッグ命令読出手段と、

前記デバッグ命令読出手段で読み出されたデバッグ命令に対応して前記OS管理情報を変更する変更手段と

を備えることを特徴とするデバッグ支援装置。

【請求項6】 前記デバッグ命令書込手段は、前記OSデバッガ側に備えられ、当該OSデバッガの要求に応じてデバッグ命令を前記共有管理情報に書き込み、

前記デバッグ命令読出手段は、前記アプリケーション側に備えられ、前記デバッグ命令書込手段で書き込まれたデバッグ命令を読み出すことを特徴とする請求項5に記載のデバッグ支援装置。

【請求項7】 前記デバッグ命令読出手段は、任意のタイミングで前記共有ファイルからデバッグ命令を読み出す請求項5または6に記載のデバッグ支援装置。

【請求項8】 前記デバッグ命令読出手段は、所定の周期で前記共有ファイルからデバッグ命令を読み出す請求項5または6に記載のデバッグ支援装置。

【請求項9】 汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、

前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記デバッグ支援装置は、

前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、

前記OSシミュレータによって前記記憶手段のOS管理情報が変更された場合、当該変更されたOS管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、

前記アプリケーションと前記OSデバッガとの間でプロセス間通信を実行する通信手段と、

前記共有ファイルに記憶された共有管理情報を読み出す読出手段とを更に備えており、

前記通信手段は、

前記OS管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記OSデバッガに送信して前記アプリケーションの実行を停止させ、

前記共有管理情報が前記読出手段によって読み出された場合、前記アプリケーションを再起動する指示を前記OSデバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されることを特徴とするデバッグ支援装置。

【請求項10】 前記通信手段は、前記アプリケーション側、前記OSデバッガ側それぞれに備えられており、

前記アプリケーションの通信手段は、前記OS管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記共有管理情報を読み出す指示を前記OSデバッガの通信手段に送信し、

前記OSデバッガの通信手段は、前記共有管理情報が前記読出手段によって読み出された場合、前記アプリケーションを再起動する指示を前記アプリケーションの通信手段に返信することを特徴とする請求項9に記載のデバッグ支援装置。

【請求項11】 汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記アプリケーションと前記OSデバッガとで共有され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する共有ファイルと、

前記OSシミュレータの要求に応じて、前記共有ファイルに記憶されたOS管理情報を更新する更新手段と、

前記OSデバッガの要求に応じて、前記更新手段で更新されたOS管理情報を読み出す読出手段と

を備えることを特徴とするデバッグ支援装置。

【請求項12】 汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、

前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有し、

前記記憶手段のOS管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記書込ステップでOS管理情報が書き込まれた共有管理情報を読み出す読出

ステップと

を含むことを特徴とするデバッグ支援方法。

【請求項 1 3】 汎用 OS 上でリアルタイム OS をシミュレートする OS シミュレータがリンクされたアプリケーションと、

前記 OS シミュレータで変更され、前記アプリケーションのタスク実行を制御する OS 管理情報を記憶する記憶手段と、

前記 OS シミュレータで変更される OS 管理情報を参照または変更する OS デバッガとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記 OS デバッガとで共有され、前記記憶手段に記憶された OS 管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルを更に有しており、

外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込ステップと、

前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込ステップで書き込まれたデバッグ命令を読み出すデバッグ命令読出ステップと、

前記デバッグ命令読出ステップで読み出されたデバッグ命令に対応して前記 OS 管理情報を変更する変更ステップと

を含むことを特徴とするデバッグ支援方法。

【請求項 1 4】 汎用 OS 上でリアルタイム OS をシミュレートする OS シミュレータがリンクされたアプリケーションと、

前記 OS シミュレータで変更され、前記アプリケーションのタスク実行を制御する OS 管理情報を記憶する記憶手段と、

前記 OS シミュレータで変更される OS 管理情報を参照または変更する OS デバッガとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記 OS デバッガとで共有され、前記記憶手段に記憶さ

れた OS 管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有し、

前記 OS シミュレータによって前記記憶手段の OS 管理情報が変更された場合、当該変更された OS 管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記アプリケーションと前記 OS デバッガとの間でプロセス間通信を実行する通信ステップと、

前記共有ファイルに記憶された共有管理情報を読み出す読出ステップとを含み、

前記通信ステップでは、

前記 OS 管理情報の変更が前記共有管理情報に前記書込ステップによって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記 OS デバッガに送信して前記アプリケーションの実行を停止させ、

前記共有管理情報が前記読出ステップで読み出された場合、前記アプリケーションを再起動する指示を前記 OS デバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されることを特徴とするデバッグ支援方法。

【請求項 15】 前記アプリケーション、OS デバッガはそれぞれ通信手段を備えており、

前記通信ステップでは、

前記 OS 管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記アプリケーションの通信手段によって前記共有管理情報を読み出す指示を前記 OS デバッガの通信手段に送信し、

前記共有管理情報が前記読出ステップによって読み出された場合、前記 OS デバッガの通信手段によって前記アプリケーションを再起動する指示を前記アプリケーションの通信手段に返信することを特徴とする請求項 14 に記載のデバッグ支援方法。

【請求項 16】 汎用 OS 上でリアルタイム OS をシミュレートする OS シミュレータがリンクされたアプリケーションと、

前記 OS シミュレータで変更される OS 管理情報を参照または変更する OS デ

バグとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記OSデバッグとで共有され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する共有ファイルを有し、

前記OSシミュレータの要求に応じて、前記共有ファイルに記憶されたOS管理情報を更新する更新ステップと、

前記OSデバッグの要求に応じて、前記更新ステップで更新されたOS管理情報を読み出す読出ステップと

を含むことを特徴とするデバッグ支援方法。

【請求項17】 汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、

前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッグとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

前記アプリケーションと前記OSデバッグとで共有され、前記記憶手段に記憶されたOS管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有しており、

前記コンピュータに、

前記記憶手段のOS管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記書込ステップでOS管理情報が書き込まれた共有管理情報を読み出す読出ステップと

を実行させるプログラムが記録された記録媒体。

【請求項18】 汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、

・前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルを更に有しており、

前記コンピュータに、

外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込ステップと、

前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込ステップで書き込まれたデバッグ命令を読み出すデバッグ命令読出ステップと、

前記デバッグ命令読出ステップで読み出されたデバッグ命令に対応して前記OS管理情報を変更する変更ステップと

を実行させるプログラムが記録された記録媒体。

【請求項19】 汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、

前記OSシミュレータで変更され、当該アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有しており、

前記コンピュータに、

前記OSシミュレータによって前記記憶手段のOS管理情報が変更された場合、当該変更されたOS管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記アプリケーションと前記OSデバッガとの間でプロセス間通信を実行する通信ステップと、

前記共有ファイルに記憶された共有管理情報を読み出す読出ステップとを実行させ、

前記通信ステップでは、

前記OS管理情報の変更が前記共有管理情報に前記書込ステップによって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記OSデバッガに送信して前記アプリケーションの実行を停止させ、

前記共有管理情報が前記読出ステップで読み出された場合、前記アプリケーションを再起動する指示を前記OSデバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されることを特徴とする記録媒体。

【請求項20】 前記アプリケーション、OSデバッガはそれぞれ通信手段を備えており、

前記通信ステップでは、

前記OS管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記アプリケーションの通信手段によって前記共有管理情報を読み出す指示を前記OSデバッガの通信手段に送信し、

前記共有管理情報が前記読出ステップによって読み出された場合、前記OSデバッガの通信手段によって前記アプリケーションを再起動する指示を前記アプリケーションの通信手段に返信することを特徴とする請求項19に記載の記録媒体。

【請求項21】 汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、

前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

・ 前記アプリケーションと前記ＯＳデバッガとで共有され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する共有ファイルを有し、
・ 前記コンピュータに、
前記ＯＳシミュレータの要求に応じて、前記共有ファイルに記憶されたＯＳ管理情報を更新する更新ステップと、
前記ＯＳデバッガの要求に応じて、前記更新ステップで更新されたＯＳ管理情報を読み出す読出ステップと
を実行させるプログラムが記録された記録媒体。

【発明の詳細な説明】

【 0 0 0 1 】

【発明の属する技術分野】

本発明は、デバッグ支援装置、デバッグ支援方法及びそのプログラムを記録した記録媒体に関し、特に、ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際に用いるデバッグ支援装置に関するものである。

【 0 0 0 2 】

【従来の技術】

近年急速に普及しつつある携帯電話などの移動体端末（以下、機器）では、通話機能、電話帳登録機能、インターネット通信など様々な機能の実現が要求されている。そのため移動体端末には、このような様々な機能を実現するためのアプリケーションが組み込まれるようになっている。

【 0 0 0 3 】

図 7 は、組み込み型のアプリケーションの開発環境の説明図である。図 7（a）は、機器にアプリケーションが組み込まれた状態を表す概念図である。アプリケーション 2 1 0 は、通常、機器専用のオペレーティングシステム（以下、専用ＯＳ；Operating System）2 2 0 a 上で動作するようになっている。アプリケーション 2 1 0 からの要求にしたがって、専用ＯＳ 2 2 0 a が、機器内のハードウェアを制御することによって様々な機能が実現されることになる。

【 0 0 0 4 】

・ 従って、このような組み込み型のアプリケーション 2 1 0 の開発は機器のハードウェアや専用 OS 2 0 a の開発と並行して進める必要がある。通常、アプリケーション 2 1 0 の組み込み先である機器の環境を想定した仮想的な環境（以下、シミュレーション環境）が構築された汎用コンピュータ上で、アプリケーション 2 1 0 の開発が汎用 OS を用いて行われる。

【 0 0 0 5 】

図 7（b）は、汎用コンピュータ上のシミュレーション環境の状態を表す説明図である。汎用コンピュータ内のハードウェアを制御する汎用 OS 2 5 0 上で、アプリケーション 2 1 0、OS シミュレータ 2 2 0、アプリケーションデバッガ 2 3 0、OS デバッガ 2 4 0 が実行される。図 7（b）では、専用 OS 2 2 0 a の動作（例えば、タスク管理動作や、イベントフラグなどのリソース管理動作）を仮想的に実行する OS シミュレータ 2 2 0 が、ライブラリという形式でアプリケーション 1 0 にリンクされている。これにより、アプリケーション 2 1 0 が組み込まれる機器のシミュレーション環境が構築されている。

【 0 0 0 6 】

また、アプリケーション 2 1 0 のデバッグは、汎用のアプリケーション開発ツール（図示せず）に含まれるアプリケーションデバッガ 2 3 0 を用いて行うようになっている。ここでデバッグとは、プログラム中のバグを取り除く作業をいい、デバッガとは、デバッグを効率よく行えるようにしたユーティリティプログラムをいう。

【 0 0 0 7 】

通常、アプリケーション 2 1 0 をデバッグする場合、図示しないメモリにおいて OS シミュレータ 2 2 0 が管理する各種情報（以下、OS 管理情報）を参照あるいは変更する場合がある。OS 管理情報には、例えば、アプリケーション 2 1 0 の実行中におけるタスクの状態や、タスクの状態を変化させるイベントの有無を 1 または 0 のビットパターンで表したイベントフラグなどがある。

【 0 0 0 8 】

しかしながら、アプリケーションデバッガ 2 3 0 は、汎用のアプリケーション

開発ツールであるため、OS管理情報を参照あるいは変更するというような機能は備えていない。そこで、最近では、アプリケーションデバッガ230だけでなく、OS管理情報を操作するためのユーティリティプログラムであるOSデバッガ240を備えたデバッグ支援システムが登場している。

【0009】

図6は、従来におけるデバッグ支援システムの一例を示す機能ブロック図である。図6に示すように、OSデバッガ240は、OSシミュレータ220とは別のプロセス空間に存在するため、OS管理情報に直接アクセスすることはできない。従って、アプリケーション210側とOSデバッガ240側の両方に通信手段C1、C2を備え、通信手段C1、C2でプロセス間通信を行うことによって、OSデバッガ240からOS管理情報を操作することが可能となる。

【0010】

また、デバッグ作業時においてバグの位置を見当付けるのを容易にするため、ブレークポイントを設定するなどしてアプリケーション210を停止させ、この時点のOS管理情報を参照することがある。

【0011】

【発明が解決しようとする課題】

しかしながら、アプリケーション210が停止した場合は、アプリケーション210とリンクしているOSシミュレータ220も停止する。すなわち、図6に示したような従来のデバッグ支援システムのようにプロセス間通信を採用した構成によれば、アプリケーション210が停止すると、OSシミュレータ220とOSデバッガ240との間で通信が行えなくなり、この時点のOS管理情報を参照できなくなり、デバッグ作業時においてバグの位置を見当付けるのが困難であった。

【0012】

ブレークポイントを設定してアプリケーション210が意識的に停止された場合だけでなく、アプリケーション210がハングアップした場合もOS管理情報を参照できず、ブレークポイントが設定された場合よりも更にバグの位置を見当付けることが困難となる。

【 0 0 1 3 】

本発明の目的は、OSシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のOS管理情報を確実に参照でき、アプリケーションのデバッグの効率を向上させることである。

【 0 0 1 4 】

【課題を解決するための手段】

上記課題を解決するために、本発明の第1の態様によれば、汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、前記記憶手段のOS管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、前記書込手段でOS管理情報が書き込まれた共有管理情報を読み出す読出手段とを備えるようになっている。

【 0 0 1 5 】

また、上記課題を解決するために、本発明の第2の態様によれば、汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルと、外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込手段と、前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込手段で書き込まれたデバッグ命令を読み出すデバッグ命令読

出手段と、前記デバッグ命令読出手段で読み出されたデバッグ命令に対応して前記OS管理情報を変更する変更手段とを備えるようになっている。

【0016】

また、上記課題を解決するために、本発明の第3の態様によれば、汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、前記OSシミュレータで変更され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する記憶手段と、前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記デバッグ支援装置は、前記アプリケーションと前記OSデバッガとで共有され、前記記憶手段に記憶されたOS管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、前記OSシミュレータによって前記記憶手段のOS管理情報が変更された場合、当該変更されたOS管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、前記アプリケーションと前記OSデバッガとの間でプロセス間通信を実行する通信手段と、前記共有ファイルに記憶された共有管理情報を読み出す読出手段とを更に備えており、前記通信手段は、前記OS管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記OSデバッガに送信して前記アプリケーションの実行を停止させ、前記共有管理情報が前記読出手段によって読み出された場合、前記アプリケーションを再起動する指示を前記OSデバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されるようになっている。

【0017】

また、上記課題を解決するために、本発明の第4の態様によれば、汎用OS上でリアルタイムOSをシミュレートするOSシミュレータがリンクされたアプリケーションと、前記OSシミュレータで変更されるOS管理情報を参照または変更するOSデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、前記アプリケーションと前記OSデバッガとで共有され、前記アプリケーションのタスク実行を制御するOS管理情報を記憶する共有ファイル

と、前記OSシミュレータの要求に応じて、前記共有ファイルに記憶されたOS管理情報を更新する更新手段と、前記OSデバッガの要求に応じて、前記更新手段で更新されたOS管理情報を読み出す読出手段とを備えるようになっている。

【0018】

【発明の実施の形態】

以下に本発明の一実施の形態を図面に従って詳細に説明する。

(第1の実施の形態)

以下、本発明のデバック支援装置における第1の実施の形態について図面を参照しながら詳細に説明する。本実施の形態においては、本発明におけるデバック支援装置が携帯電話などの移動体端末のアプリケーションのデバック作業に適用された場合について説明する。

【0019】

図1は、本発明を適用したデバック支援装置の機能ブロック図である。本実施の形態に係るデバック支援装置110は、例えば、アプリケーション10、OSシミュレータ20、記憶手段90、OSデバッガ40、共有ファイルF、表示制御部121、インターフェイス部122及び図示しないアプリケーションデバックとを備えている。本デバック支援装置110においては、汎用OS（図示せず）、OSシミュレータ20及びOSデバッガ40が協働して、アプリケーション10が組み込まれる機器のシミュレーション環境が構築されている。

【0020】

アプリケーション10は、移動体端末の様々な業務処理を行うためのプログラムである。アプリケーション10は、ライブラリという形式でOSシミュレータ20がリンクされており、共有管理情報書き込み手段12を備えている。

【0021】

OSシミュレータ20は、システムコール処理手段21とOS管理情報変更手段22とを備えている。OSシミュレータ20は、タスク管理やイベントフラグなどのリソース管理といった専用OS（図示せず）の動作を擬似的に実行する。システムコール処理手段21は、アプリケーションデバッガ（図示せず）によってアプリケーション10のタスクが実行されたときに発行されるシステムコール

を処理し、システムコールの処理によるOS管理情報を変更する命令をOS管理情報変更手段22に通知する。ここで、OS管理情報とは、OSシミュレータが管理するデータであり、例えばアプリケーション10実行中のタスクの状態をあらわすタスク実行状況やタスクの状態を変化させるイベントの有無を1又は0のビットパターンで表したイベントフラグなどがある。

【0022】

OS管理情報変更手段22は、システムコール処理手段21からのOS管理情報の変更命令に従って、後述する記憶手段90に格納されているOS管理情報OS2を変更し、更にOS管理情報OS2の変更内容を共有管理情報書き込み手段12に通知する。

【0023】

共有管理情報書き込み手段12は、OS管理情報変更手段22から通知されたOS管理情報OS2の変更内容を後述する共有ファイルFに書き込む。

【0024】

記憶手段90は、RAM (Random Access Memory) で構成される。記憶手段90には、OS管理情報変更手段22で変更されるOS管理情報OS2が記憶される。図1においては、一例として、アプリケーション10のタスクAに対応するOS管理情報が記憶されている。OS管理情報OS2には、イベントフラグEF2が含まれている。イベントフラグEF2は、タスクAの状態を変化させるイベントの有無を1又は0のビットパターンで表したものである。

【0025】

共有ファイルFは、アプリケーション10やOSデバッガ40によって共有されるハードディスク内のメモリ領域に記憶され、各プロセスで共有される共有管理情報SS2を含むデータ群を示す。共有ファイルFには、記憶手段90に記憶されたOS管理情報と同一のデータが含まれたデータである共有管理情報SS2が格納される。アプリケーション10のタスク毎の実行に伴う、OSシミュレータ20によるOS管理情報OS2の変更に同期して、OS管理情報OS2の変更が共有ファイルFの共有管理情報SS2に反映されるようになっている。これに

より、アプリケーション 1 0、OS デバッガ 4 0 それぞれのプロセスで、共有ファイル F を介して OS 管理情報 OS 2 の共有化が実現できる。また、共有管理情報 SS 2 はアプリケーション 1 0 と OS デバッガ 4 0 とで共有されるので、ハードディスクなどの不揮発性メモリ上に記憶することが望ましい。FD などの外部記録媒体であってもよい。共有管理情報 SS 2 が消去されることがなく、アプリケーション 1 0 と OS デバッガ 4 0 とでデータの共有化が確実にできるためである。

【 0 0 2 6 】

図 1 においては、一例として、アプリケーション 1 0 のタスク A に対応する OS 管理情報と同一のデータを含む共有管理情報 SS 2 が記憶されている。共有管理情報 SS 2 には、イベントフラグ EF 3 が含まれており、イベントフラグ EF 3 は記憶手段 9 0 のイベントフラグ EF 2 に対応する。つまり、記憶手段 9 0 の OS 管理情報 OS 2 と共有ファイル F の共有管理情報 SS 1 とは完全に同期されるようになっている。

【 0 0 2 7 】

OS 管理情報 OS 2 を操作するためのユーティリティプログラムである OS デバッガ 4 0 は、ユーザインターフェイス 4 1、メイン処理手段 4 2、共有管理情報読み出し手段 4 3、表示処理手段 4 4 とを備えている。

【 0 0 2 8 】

共有管理情報読み出し手段 4 3 は、メイン処理手段 4 2 の命令により、共有管理情報 SS 2 を読み出し、読み出された共有管理情報 SS 2 を表示するよう表示処理手段 4 4 へ命令する。表示処理手段 4 4 は、共有管理情報読み出し手段 4 3 によって読み出された共有管理情報 SS 2 を表示制御部 1 2 1 で映像信号に変換し、ディスプレイの画面（図示せず）に表示する。

【 0 0 2 9 】

ユーザインターフェイス 4 1 は、ユーザによって操作部（図示せず）からの入力信号がインターフェイス部 1 2 2 を介して通知された場合、共有ファイル F 1 に格納された共有管理情報 SS 2 を表示させる処理をメイン処理手段 4 2 へ命令する。メイン処理手段 4 2 は、ユーザインターフェイス 4 1 の命令により、共有

管理情報読み出し手段43へ共有管理情報SS2を読み出す命令をする。

【0030】

ここで、本実施の形態では、図示しない中央演算処理装置（CPU）を含む制御部が、本発明のプログラムにしたがってCPU以外の各回路と協働して実行される制御動作を実現している。しかし、以下では説明の便宜上、CPUが関係する制御動作を、アプリケーション10などのプログラムが直接的に制御しているものとしてその説明を行う。

【0031】

ここでは、例として、タスクAが実行され、システムコールである「set_flg」、「clr_flg」が発行される場合を用いて説明する。これらのシステムコールによってイベントフラグの任意のビットがセットされ、他のタスクの状態を待ち状態としたり、待ち状態を解除して実行可能状態と変更できる。システムコール「set_flg」が発行されるとOS管理情報OS2のイベントフラグEF2の値が0から1に変更される。また、システムコール「clr_flg」が発行されるとOS管理情報OS2のイベントフラグEF2の値が1から0に変更されるようになっている。

【0032】

まず、アプリケーション10側の処理の流れを説明する。アプリケーションデバッガ（図示せず）によってアプリケーション10のタスクAが実行されて、システムコール「set_flg」が発行される。発行されたシステムコール「set_flg」がシステムコール処理手段21で処理され、OS管理情報OS2のイベントフラグEF2の値を0から1に変更する命令がOS管理情報変更手段22に通知される。

【0033】

続いて、OS管理情報変更手段22は、OS管理情報OS2のイベントフラグEF2の値を0から1に変更し、イベントフラグEF2を0から1に変更した内容を共有管理情報書き込み手段12に通知する。

【0034】

共有管理情報書き込み手段12は、OS管理情報OS2の変更内容を共有ファイ

ルFのイベントフラグEF3に書き込む。具体的には、共有管理情報SS2のイベントフラグEF3の値を0から1に変更する。

【0035】

OS管理情報OS2の変更内容が共有ファイルFに書き込まれた後、アプリケーション10の動作が停止される。アプリケーション10の動作が停止された時点のイベントフラグの状況を参照したい場合は、ユーザによってキーボードなどの操作部が操作され、操作部から入力された入力信号がインターフェイス部122を介してユーザインターフェイス41に通知される。ユーザインターフェイス41からの指示に応じて、メイン処理手段42は、共有管理情報SS2のイベントフラグEF3を読み出すように共有管理情報読み出し手段43に命令する。

【0036】

共有管理情報読み出し手段43は、メイン処理手段42からの命令に従って、共有管理情報SS2のイベントフラグEF3を読み出す。読み出されたイベントフラグEF3の情報は共有管理情報読み出し手段43から表示処理手段44へ通知され、表示制御部121で映像信号に変更されてディスプレイの画面（図示せず）に表示される。

【0037】

この際、イベントフラグEF3が1であることをユーザが視覚的に理解できるように画面上に表示させることによって、ユーザはアプリケーション10と汎用OS（図示せず）との処理の連携、アプリケーション10とOSデバッガ40との処理間で同期がとられたことを容易に把握することができる。

【0038】

以上のように、OS管理情報変更手段22によるOS管理情報OS2の変更に同期して、OS管理情報OS2の変更が共有ファイルFの共有管理情報SS2に反映されるので、アプリケーション10が停止した場合でもOS管理情報OS2を容易に参照することができる。これにより、効率の良いデバッグが実行することができるようになる。

【0039】

なお、イベントフラグの値を共有ファイルFから読み出すタイミングは、「イ

イベントフラグの参照」が指示されたときに限定されるものではない。例えば、共有管理情報読み出し手段 4 3 に対して周期的にイベントフラグ E F 3 を含む共有管理情報 S S 2 の読み出しを指示する手段を備えた構成としてもよい。これにより、O S 管理情報 O S 2 の時間的遷移を知ることができるため、予期しないイレギュラーなアプリケーション 1 0 の実行状態を容易に発見することができ、デバッグの効率が向上されることになる。

【 0 0 4 0 】

以上、システムコール「set_flg」の場合について説明したが、システムコール「clr_flg」の場合でも同様に O S 管理情報 O S 1 のイベントフラグ E F 2 が 1 から 0 に変更されたことをユーザに視覚的に伝えることができる。

【 0 0 4 1 】

また、O S 管理情報 O S 2 に含まれるデータの種別毎に共有ファイル F 内のメモリ領域を予め割り当てておけば、イベントフラグだけでなく、他の O S 管理情報（タスクの状態・セマフォの状態・メールボックスの状態など）も同様の手順で参照できることはいうまでもない。

【 0 0 4 2 】

（第 2 の実施の形態）

以下、本発明のデバック支援装置における第 2 の実施の形態について図面を参照しながら詳細に説明する。本実施の形態においては、第 1 の実施の形態において、画面上で参照された O S 管理情報 O S 2 の値が不正である場合、O S 管理情報 O S 2 を O S デバッガ 4 0 側から変更できるようになっている。

【 0 0 4 3 】

デバッグ中に不正な値を有する O S 管理情報 O S 2 が発生した場合、アプリケーション 1 0 にバグが存在するので、そのバグを検証して取り除く必要がある。しかし、一般的には、デバッグの効率を向上させるために、バグが発生するたびに個別に検証するのではなくアプリケーション 1 0 の動作全体について検証が行われる。本デバック支援装置 1 2 0 では、画面上で参照された O S 管理情報の値が不正である場合、O S 管理情報 O S 2 中の不正な値を有するデータを O S デバ

ッガ 4 0 側から変更できる。これにより、アプリケーション 1 0 のプログラムを修正することなく、アプリケーション 1 0 の動作を継続させてデバッグの効率を向上させることが可能となる。

【 0 0 4 4 】

図 2 は、本実施の形態に係るデバッグ支援装置 1 2 0 の機能ブロック図である。本実施の形態に係るデバッグ支援装置 1 2 0 は、第 1 の実施の形態のデバッグ支援装置 1 1 0 と同様に、例えば、アプリケーション 1 0、OS シミュレータ 2 0、記憶手段 9 0、OS デバッガ 4 0、共有ファイル F、表示制御部 1 2 1、インターフェイス部 1 2 2 及び図示しないアプリケーションデバッガとを備えている。本デバッグ支援装置 1 2 0 は、第 1 の実施の形態のデバッグ支援装置 1 1 0 と同様に、図示はしないが、アプリケーション 1 0 側に共有管理情報書き込み手段 1 2、OS デバッガ 4 0 側に共有管理情報読み出し手段 4 3 を備えており、共有ファイル F へのデータの読み書き処理は可能である。以下、本実施の形態に係るデバッグ支援装置 1 2 0 と実施の形態 1 に係るデバッグ支援装置 1 1 0 との構成上の差異について説明する。

【 0 0 4 5 】

まず、アプリケーション 1 0 は、OS シミュレータ 2 0 以外に、デバッグタスク起動手段 1 3、システムコール情報生成手段 1 4、デバッグ命令読み出し手段 1 5、タイマ 1 6 を更に備えている。

【 0 0 4 6 】

デバッグ命令読み出し手段 1 5 は、共有ファイル F の共有管理情報 S S 2 に含まれるデバッグ命令機能コードをタイマ 1 6 を用いて周期的に読み出す。ここでデバッグ命令機能コードとは、OS 管理情報 O S 2 のデータを OS シミュレータ 2 0 によって変更するためのシステムコールを特定するためのデータである。

【 0 0 4 7 】

システムコール情報生成手段 1 4 は、デバッグ命令機能コードに対応するシステムコールの実行タスクを特定するための情報（以下、システムコール情報）を生成する。デバッグタスク起動手段 1 3 は、システムコール情報生成手段 1 4 で生成されたシステムコール情報に応じてデバッグタスク 1 7 を起動する。起動さ

れたデバッグタスク17がOSシミュレータ20で実行処理されて、記憶手段90のOS管理情報OS2が変更されるようになっている。

【0048】

また、OSデバッガ40は、ユーザインタフェース41、メイン処理手段42、表示処理手段44以外に、デバッグ命令書き込み手段45を更に備えている。

【0049】

デバッグ命令書き込み手段45は、ユーザからのOS管理情報OS2の修正指示（以下、デバッグ命令）に応じて、デバッグ命令に対応するデバッグ命令機能コードを共有ファイルFに書き込む。

【0050】

以下、OSデバッガ40から、最終的にOS管理情報OS2が変更されるまでの処理の流れを説明する。ここでは、第1の実施の形態で記載したように、タスクAによってシステムコール「set_flg」が実行された後、参照された共有ファイルFのイベントフラグEF3が不正な値「0」であった場合について説明する。この場合、最終的にOS管理情報OS2のイベントフラグEF2を「1」に変更する必要がある。

【0051】

まず、OSデバッガ側からの処理の流れを説明する。イベントフラグEF3が不正な値「0」であることがユーザによって確認された場合、OS管理情報OS2のイベントフラグEF2を「1」に変更するためのデバッグ命令が操作部から入力される。入力されたデバッグ命令がインターフェイス部122を介してOSデバッガ40のユーザインターフェイス41に通知される。続いて、ユーザインターフェイス41からの指示に応じて、メイン処理手段42は、システムコール「set_flg」を特定するためのデバッグ命令機能コード（-48）を共有管理情報SS2に書き込むようにデバッグ命令書き込み手段45に命令する。

【0052】

メイン処理手段42からの命令に応じて、デバッグ命令書き込み手段45によって、共有管理情報SS2中にデバッグ命令機能コードCOが書き込まれる。

【0053】

・続いて、アプリケーション 10 側の処理について説明する。デバッグ命令機能コード C0 がデバッグ命令読み出し手段 15 によって読み出され、システムコール情報生成手段 14 に渡される。ここでデバッグ命令読み出し手段 15 はタイマを用いて周期的に共有ファイル F のデバッグ命令機能コードを読み出すが、読み出し周期はデバッグ命令機能コードが書き込まれる周期より短いのが望ましい。これにより、デバッグ命令機能コードの読み出しの取りこぼしを確実に防ぐことができるようになる。

【 0 0 5 4 】

デバッグ命令機能コード C0 に対応する、例えばパラメータなどのシステムコール情報がシステムコール情報生成手段 14 で生成され、デバッグタスク起動手段 13 に渡される。渡されたシステムコール情報に従って、デバッグタスク起動手段 13 は、システムコール「set_flg」を発行するデバッグタスク 17 を起動する。

【 0 0 5 5 】

起動されたデバッグタスク 17 によって、OS シミュレータ 20 に対してシステムコール「set_flg」が発行され、OS 管理情報 OS 2 のイベントフラグ EF 2 が「0」から「1」に変更される。イベントフラグ EF 2 の変更後、イベントフラグ EF 2 待ちの状態にあるタスク B が実行状態に遷移される。

【 0 0 5 6 】

以上のように、本実施の形態によれば、何らかのバグが存在することによって OS 管理情報 OS 2 が不正な値となった場合でも、この OS 管理情報 OS 2 を OS デバッガ 40 から変更することができる。また、このような変更機能を利用すれば、アプリケーション 10 のプログラムを修正することなく、割り込みが発生した状況をつくることも可能である。

【 0 0 5 7 】

なお、システムコール情報を指定する方法は、システムコール「set_flg」を指示するパラメータだけでなく、システムコール「set_flg」を指示するメッセージの受け渡しであってもよい。また、共有ファイル F を用いてシステムコール情報を指定してもよい。具体的には、デバッグタスク 17 が直接、

共有ファイルFの中のデバッグ命令機能コードを参照してシステムコール情報を生成するようにすればよい。

【 0 0 5 8 】

また、OSデバッガ40からデバッグ命令を指示することとしているが、本発明はこれに限定されるものではない。すなわち、予めデバッグ命令を共有ファイルFに格納しておき、このデバッグ命令をデバッグ命令読み出し手段15が周期的に（或いは、任意のタイミングで）読み出すようにしてもよい。このようにすれば、OSデバッガ40を用いることなくOS管理情報OS2を変更できる。

【 0 0 5 9 】

また、デバッグ命令を実行する手段はデバッグタスク17に限定されるものではない。すなわち、ここでは、デバッグタスク17がシステムコールを発行することによってデバッグ命令が実行される動作を例示しているが、システムコールには、タスクからのみ発行可能・割り込みハンドラからのみ発行可能・この両方から発行可能という種別があるため、この種別に応じた実行手段を選択するようにしている。

【 0 0 6 0 】

更に、上記の説明では、デバッグタスク17の状態遷移について特に言及していないが、システムコールを発行し終えたデバッグタスク17は、消滅しても、常駐しても（すなわち、次のデバッグ命令が共有ファイルFに書き込まれるまで待ち状態となっても）、いずれであってもかまわない。

（第3の実施の形態）

以下、本発明のデバック支援装置における第3の実施の形態について図面を参照しながら詳細に説明する。

【 0 0 6 1 】

本実施の形態におけるデバック支援装置は他の実施の形態と同様に共有ファイルを備えるとともに、更に、アプリケーションとOSデバッガとの間でプロセス間通信を実行する通信手段を備えていることを特徴とする。この通信手段により、OSデバッガは、OSシミュレータによってOS管理情報が変更された場合に漏れなくOS管理情報の変更内容を参照することができる。具体的には、OSデ



バッガによってタイマを用いて共有ファイルからデータを周期的に読み出す場合、共有ファイルからデータを読み出す周期と、OS管理情報がOSシミュレータによって変更される周期を同期させることができる。

【0062】

図3は、本実施の形態におけるデバッグ支援装置の機能ブロック図である。本実施の形態に係るデバッグ支援装置1は、例えば、アプリケーション60、OSシミュレータ70、記憶手段100、OSデバッガ80、共有ファイルF1、表示制御部91、インターフェイス部92及び図示しないアプリケーションデバッガとを備えている。本デバッグ支援装置1においては、汎用OS（図示せず）、OSシミュレータ70及びOSデバッガ80が協働して、アプリケーション60が組み込まれる機器のシミュレーション環境が構築されている。

【0063】

アプリケーション60は、移動体端末の様々な業務処理を行うためのプログラムである。アプリケーション60は、ライブラリという形式でOSシミュレータ70がリンクされており、共有管理情報書き込み手段62とアプリケーション通信手段C1とを備えている。

【0064】

OSシミュレータ70は、システムコール処理手段71とOS管理情報変更手段72とを備えている。OSシミュレータ70は、タスク管理やイベントフラグなどのリソース管理といった専用OS（図示せず）の動作を擬似的に実行する。システムコール処理手段71は、アプリケーションデバッガ（図示せず）によってアプリケーション60のタスクが実行されたときに発行されるシステムコールを処理し、システムコールの処理によるOS管理情報を変更する命令をOS管理情報変更手段72に通知する。ここで、OS管理情報とは、OSシミュレータが管理するデータであり、例えばアプリケーション60実行中のタスクの状態をあらわすタスク実行状況やタスクの状態を変化させるイベントの有無を1又は0のビットパターンで表したイベントフラグなどがある。

【0065】

OS管理情報変更手段72は、システムコール処理手段71からのOS管理情

報の変更命令に従って、後述する記憶手段100に格納されているOS管理情報OS1を変更し、更にOS管理情報OS1の変更内容を共有管理情報書き込み手段62に通知する。

【0066】

共有管理情報書き込み手段62は、OS管理情報変更手段72から通知されたOS管理情報OS1の変更内容を後述する共有ファイルF1に書き込む。

【0067】

アプリケーション通信手段C1は、アプリケーション60とOSデバッガ80とのプロセス間通信を実行する。OS管理情報変更手段72から通知されたOS管理情報OS1の変更内容が共有管理情報書き込み手段62によって共有ファイルF1に書き込まれたことをOSデバッガ80に通信する。

【0068】

記憶手段100は、RAM (Random Access Memory) で構成される。記憶手段100には、OS管理情報変更手段72で変更されるOS管理情報OS1が記憶される。図3においては、一例として、アプリケーション60のタスクCに対応するOS管理情報が記憶されている。OS管理情報OS1には、イベントフラグEFやタスク実行状態TSが含まれている。イベントフラグEFはタスクCの状態を変化させるイベントの有無を1又は0のビットパターンで表したものである。また、タスク実行状態TSはアプリケーション60実行中のタスクCの状態を表したものである。図3の例では、タスクCが実行状態であることが示されている。

【0069】

共有ファイルF1は、他の実施の形態の共有ファイルFと同様にアプリケーション60とOSデバッガ80とで共有されている。共有ファイルF1には、記憶手段100に記憶されたOS管理情報と同一のデータが含まれたデータである共有管理情報SS1が格納される。図3においては、一例として、アプリケーション60のタスクCに対応するOS管理情報と同一のデータを含む共有管理情報SS1が記憶されている。共有管理情報SS1には、イベントフラグEF1やタスク実行状態TS1が含まれている。イベントフラグEF1は記憶手段のイベント

フラグEFに対応する。また、タスク実行状態TS1は記憶手段のタスク実行状態TSに対応する。

【0070】

また、共有ファイルF1の共有管理情報SS1には、アプリケーション60のタスク毎の実行に伴う、OSシミュレータ70によるOS管理情報OS1の変更と同期してOS管理情報OS1の変更が反映されるようになっている。更に、アプリケーション通信手段C1からの変更通知をOSデバッガ80が受信した場合に、OS管理情報OS1の変更が反映された共有管理情報SS1がOSデバッガ80によって読み出されるようになっている。つまり、記憶手段100のOS管理情報OS1と共有ファイルF1の共有管理情報SS1とは完全に同期される。

【0071】

OS管理情報を操作するためのユーティリティプログラムであるOSデバッガ80は、ユーザーインターフェイス81、メイン処理手段82、共有管理情報読み出し手段83、表示処理手段84及びOSデバッガ通信手段C2とを備えている。

【0072】

OSデバッガ通信手段C2はアプリケーション60とのプロセス間通信を実行する。アプリケーション通信手段C1からOS管理情報OS1の変更内容が共有ファイルF1の共有管理情報SS1に書き込まれたという通知を受信した場合に、共有管理情報読み出し手段83に共有管理情報SS1を読み出す命令をする。

【0073】

共有管理情報読み出し手段83は、OSデバッガ通信手段C2の命令により、共有管理情報SS1を読み出し、読み出された共有管理情報SS1を表示するよう表示処理手段84へ命令する。表示処理手段84は共有管理情報読み出し手段83によって読み出された共有管理情報SS1を表示制御部91で映像信号に変換してディスプレイの画面（図示せず）に表示する。

【0074】

ユーザーインターフェイス81は、ユーザによって操作部（図示せず）からの入力信号がインターフェイス部92を介して通知された場合、共有ファイルF1

に格納された共有管理情報 S S 1 を表示させる処理をメイン処理手段 8 2 へ命令する。メイン処理手段 8 2 は、ユーザーインターフェイス 8 1 の命令により、共有管理情報読み出し手段 8 3 へ共有管理情報 S S 1 を読み出す命令をする。

【 0 0 7 5 】

ここで、本実施の形態では、図示しない中央演算処理装置 (C P U) を含む制御部が、本発明のプログラムにしたがって C P U 以外の各回路と協働して実行される制御動作を実現している。しかし、以下では説明の便宜上、C P U が関係する制御動作を、アプリケーション 6 0 などのプログラムが直接的に制御しているものとしてその説明を行う。

【 0 0 7 6 】

以下に、アプリケーション 6 0、OS シミュレータ 7 0 及び OS デバッガ 8 0 が備える各手段の具体的な処理の流れを図面を用いて説明する。図 4 は、デバッグ支援装置 1 の各手段の処理の流れを示すフローチャートである。

【 0 0 7 7 】

ここでは、例として、タスク C が実行され、システムコールである「set_flg」が発行される場合を用いて説明する。イベントフラグの任意のビットがセットされることにより、他のタスクを待ち状態としたり、待ち状態を解除して実行可能状態としたりすることができる。システムコール「set_flg」が発行されると OS 管理情報 O S 1 のイベントフラグ E F の値が 0 から 1 に変更される。システムコール「clr_flg」が発行されると OS 管理情報 O S 1 のイベントフラグ E F の値が 1 から 0 に変更されるようになっている。

【 0 0 7 8 】

まず、アプリケーション 6 0 側の処理の流れを説明する。アプリケーションデバッガ (図示せず) によってアプリケーション 6 0 のタスク C が実行されて、システムコール「set_flg」が発行される (ステップ S 6 0)。発行されたシステムコール「set_flg」がシステムコール処理手段 7 1 で処理され (ステップ S 6 1)、OS 管理情報 O S 1 のイベントフラグ E F の値を 0 から 1 に変更する命令が OS 管理情報変更手段 7 2 に通知される。

【 0 0 7 9 】

続いて、OS管理情報変更手段72は、OS管理情報OS1のイベントフラグEFの値を0から1に変更し（ステップS62）、イベントフラグEFを0から1に変更した内容を共有管理情報書き込み手段62に通知する。

【0080】

共有管理情報書き込み手段62は、OS管理情報OS1の変更内容を共有ファイルF1のイベントフラグEF1に書き込む（ステップS63）。具体的には、共有管理情報SS1のイベントフラグEF1の値を0から1に変更する。

【0081】

共有管理情報SS1のイベントフラグEF1が変更された場合、アプリケーション通信手段C1は、共有管理情報SS1の変更された共有管理情報SS1を読み出す指示（以下、変更通知）をOSデバッグ通信手段C2へ送信する（ステップS64）。その後、アプリケーション通信手段C2によって、アプリケーション60の動作が停止される（ステップS65）。ここで、アプリケーション通信手段C1からOSデバッグ通信手段C2への変更通知は、共有管理情報SS1が変更されたことを示すRAM上のフラグを介して実行してもよいし、OSの通信機能を利用してもよい。

【0082】

アプリケーション60側では、アプリケーション60の実行が停止されている間、アプリケーション通信手段C1はOSデバッグ通信手段C2からの返信があるか否かを判断する（ステップS66）。OSデバッグ通信手段C2からの返信がなければ（ステップS66；No）、アプリケーション60の動作は再開されず待ち状態となる。つまり、OSデバッグ通信手段C2からの返信があるまでアプリケーション60の次のタスクが実行されず、待ち状態が継続される。

【0083】

続いて、OSデバッグ80側の処理の流れを説明する。OSデバッグ通信手段C2は、ステップS64でアプリケーション通信手段C1から送信された変更通知を受信する（ステップS81）。OSデバッグ通信手段C2は、共有管理情報読み出し手段83へ共有管理情報SS1のイベントフラグEF1を読み出すように命令する。

【 0 0 8 4 】

共有管理情報読み出し手段 8 3 は、OS デバッガ通信手段 C 2 からの命令に従って共有管理情報 S S 1 のイベントフラグ E F 1 を読み出す（ステップ S 8 2）。読み出されたイベントフラグ E F 1 の情報は共有管理情報読み出し手段 8 3 から表示処理手段 8 4 へ通知され、表示制御部 9 1 で映像信号に変更されてディスプレイの画面（図示せず）に表示される（ステップ S 8 3）。

【 0 0 8 5 】

この際、イベントフラグ E F 1 が 1 であることをユーザが視覚的に理解できるように画面上に表示させることによって、ユーザはアプリケーション 6 0 と OS（図示せず）との処理の連携、アプリケーション 6 0 と OS デバッガ 8 0 との処理間で同期がとられたことを容易に把握することができる。

【 0 0 8 6 】

イベントフラグ E F 1 が表示された後、OS デバッガ通信手段 C 2 はアプリケーション 6 0 を再起動する指示をアプリケーション通信手段 C 1 へ返信する（ステップ S 8 4）。ここで、OS デバッガ通信手段 C 2 からアプリケーション通信手段 C 1 への返信は、共有管理情報 S S 1 のイベントフラグ E F 1 が表示されたことを示す RAM 上のフラグを介して実行してもよいし、OS の通信機能を利用してもよい。

【 0 0 8 7 】

続いて、アプリケーション 6 0 側の処理の流れに説明を戻す。アプリケーション通信手段 C 1 は OS デバッガ通信手段 C 2 からの返信を受信した場合（ステップ S 6 6 ; Y e s）、アプリケーションの動作が再開される（ステップ S 6 7）。その後、再びステップ S 6 0 に戻りアプリケーションデバッガによって、タスク C の次のタスク D が実行されて、ステップ S 6 0 からステップ S 6 7 の処理が繰り返される。

【 0 0 8 8 】

アプリケーション通信手段 C 1 と OS デバッガ通信手段 C 2 との間で双方向で通信が実行されるまでアプリケーション 6 0 の動作が停止されるので、OS デバッガ 8 0 の共有管理情報読み込み手段 8 3 によって共有管理情報 S S 1 が読み込

まれるまでOSシミュレータ70によってOS管理情報OS1が変更されることがなくなる。これにより、OSデバッガ80が、OS管理情報OS1の変更を取りこぼしなく取得することができ、アプリケーション60と別プロセス空間に存在するOSデバッガ80で、アプリケーション60に同期してデータを共有することができる。

【0089】

以上、システムコール「set_flg」の場合について説明したが、システムコール「clr_flg」の場合でも同様にOS管理情報OS1のイベントフラグEFが1から0に変更されたことをユーザに視覚的に伝えることができる。

【0090】

また、現時点で実行されているタスク（以下、自タスク）の実行状態を変更する場合にも図4に示す処理を適用できる。例えば、自タスクCを「実行状態」から「待ち状態」にするシステムコール「slp_task」、自タスクC以外のタスクやタスク以外が発行したシステムコールであり、自タスクを「待ち状態」から「実行可能状態」にするシステムコール「wup_task」の場合について説明する。

【0091】

例えば、システムコール「slp_task」がシステムコール処理手段71で処理された後（ステップS61）、OS管理情報OS1のタスク実行状態TSが「実行状態」から「待ち状態」に変更される（ステップS62）。次に、共有ファイルF1のタスク実行状態TS1が、同様に「実行状態」から「待ち状態」に変更される（ステップS64）。

【0092】

アプリケーション通信手段C1からの変更通知がOSデバッガ通信手段C2へ送信された後（ステップS65）、アプリケーション60は待ち状態となる（ステップS66）。OSデバッガ80側では、OSデバッガ通信手段C2がアプリケーション通信手段C1からの変更通知を受信した後（ステップS81）、共有ファイルF1のタスク実行状態TS1が読み出されてディスプレイの画面に表示される（ステップS83）。

【0093】

次に、OSデバッグ通信手段C2は、アプリケーション通信手段C1へ返信し（ステップS84）、アプリケーション通信手段C1で返信が受信された場合（ステップS66；Yes）、アプリケーション60の動作が再開される（ステップS67）。

【0094】

以上、システムコール「sleep_task」の場合について説明したが、システムコール「wakeup_task」の場合でも同様にOS管理情報OS1のタスク実行状態TSが「待ち状態」から「実行可能状態」に変更されたことをユーザに視覚的に伝えることができる。

【0095】

また、1つの自タスクの実行状態の変化を対象にしたが、アプリケーション60の全てのタスクに対応するタスク実行状態を共有ファイルF1の共有管理情報SS1に記憶しておいてもよい。これにより、全てのタスクの状態をユーザが視覚的に把握することができるようになる。

【0096】

また、図4の処理の流れの説明においては、OSシミュレータ70によるOS管理情報OS1の変更に同期して、共有ファイルF1の共有管理情報SS1が変更されるごとにその変更された共有管理情報SS1が画面に表示されるようになっていたが、適宜変更可能である。

【0097】

例えば、共有管理情報SS1の変更履歴をタイマを用いて周期的に共有ファイルSS1に記憶しておき、ユーザからの指示があった場合にディスプレイの画面に表示するようにしてもよい。具体的には、ユーザがOS管理情報OS1の変更をディスプレイの画面上で確認する場合、ユーザによってキーボードなどの操作部が操作されると共有ファイルF1に記憶された変更履歴が画面上に読み出されるようにすればよい。この場合、操作部から入力された入力信号がインターフェイス部92を介してユーザーインターフェイス81に通知され、メイン処理手段83の指示にしたがって共有管理情報読み込み手段83が共有ファイルF1から

変更履歴を読み出すようにすればよい。

【 0 0 9 8 】

例えば、移動体端末としての機能を実現するアプリケーション 6 0 では、通話、電話帳登録、インターネット通信など様々な種類のイベント処理が考えられる。このような複数のイベントが連続的に処理されるリアルタイム処理が要求されるアプリケーション 6 0 においては、全てのタスク間での状態遷移の相関を認識する必要がある。このような場合に、全てのタスクに対応する共有管理情報 S S 1 の変更履歴を共有ファイル F 1 に記憶しておけば、全てのタスク間の状態遷移やイベントフラグなどの履歴や、アプリケーション 6 0 と O S との連携処理におけるエラー等を画面上で容易にユーザが確認できるので、デバッグ作業の効率が向上される。

【 0 0 9 9 】

以下に、共有ファイル F 1 から複数のタスクの状態遷移が画面に表示された例について説明する。図 5 は、複数のタスクの状態遷移が表示された画面の一例を示す図である。

【 0 1 0 0 】

図 5 において、D P はディスプレイの画面を示す。S C は、アプリケーション通信手段 C 1 と O S デバッグ通信手段 C 2 との間で通信が実行されたことを表すマーク群である。S T 1 は、イベント A に対応するタスクの状態遷移を表すマーク群である。以下同様に、S T 2 はイベント B に対応するタスクの状態遷移を、S T 3 はイベント C に対応するタスクの状態遷移を表すマーク群である。

【 0 1 0 1 】

各タスクの状態遷移 S T 1、S T 2、S T 3 中のマーク M 1 は、イベントに対応するタスクが「実行可能状態」であったことを示す。つまり、O S シミュレータ 7 0 による O S 管理情報 O S 1 中のタスク実行状態の変更に応じて、共有ファイル F 1 の共有管理情報 S S 1 中のタスク実行状態が「実行可能状態」に変更されたことを示すマークである。また、各タスクの状態遷移 S T 1、S T 2、S T 3 中のマーク M 2 は、イベントに対応するタスクが「待ち状態」であったことを示す。つまり、O S シミュレータ 7 0 による O S 管理情報 O S 1 中のタスク実行

状態の変更に応じて、共有ファイル F 1 の共有管理情報 S S 1 中のタスク実行状態が「待ち状態」に変更されたことを示す。

【 0 1 0 2 】

S C 1 は、O S シミュレータ 7 0 による O S 管理情報 O S 1 中のタスク実行状態の変更に応じて、共有ファイル F 1 の共有管理情報 S S 1 中のタスク実行状態が「実行可能状態」に変更され、アプリケーション通信手段 C 1 と O S デバッガ C 2 との間で通信が実行されたことを示すマークである。

【 0 1 0 3 】

図 5 に示す例では、各イベントごとにそれぞれ、タスクの実行状態遷移、アプリケーション通信手段 C 1 と O S デバッガ C 2 との間の通信履歴とが図 5 中に表示時間軸に沿って表示されている。例えば、イベント A の最初の 5 つのタスクは「実行可能状態」であり、イベント B、C の最初の 5 つのタスクは「待ち状態」である。また、その間、アプリケーション通信手段 C 1 と O S デバッガ C 2 との間で通信が確実に実行され、アプリケーション 6 0 側による O S 管理情報 O S 1 の変更を、O S デバッガ 7 0 が同期して読み込むことができたことが示されている。

【 0 1 0 4 】

つまり、図 5 に示す画面例では、イベント A、B、C における、アプリケーション通信手段 C 1 と O S デバッガ C 2 との通信履歴、各イベントのタスクの実行状態遷移が視覚的に示されている。これにより、アプリケーション 6 0 で実行される各イベントの状態遷移、O S との連携などの不具合などをユーザは視覚的に容易に見極めることができるので、デバッグ作業の効率化を図ることが可能になる。

【 0 1 0 5 】

また、O S デバッガ 8 0 によってタイマを用いて共有ファイルからデータを周期的に読み出す場合でも、共有ファイルからデータを読み出す周期と、O S 管理情報が O S シミュレータによって変更される周期を確実に同期させることができる。

【 0 1 0 6 】

・以上、説明した通り、アプリケーション通信手段 C 1 と O S デバッガ通信手段 C 2 との間で双方向で通信が実行されるまでアプリケーション 6 0 の動作が停止されるので、O S デバッガ 8 0 の共有管理情報読み込み手段 8 3 によって共有管理情報 S S 1 が読み込まれるまで O S シミュレータ 7 0 によって O S 管理情報 O S 1 が変更されることがなくなる。これにより、O S デバッガ 8 0 が、O S 管理情報 O S 1 の変更を取りこぼしなく取得することができ、アプリケーション 6 0 と別プロセス空間に存在する O S デバッガ 8 0 で、アプリケーション 6 0 に同期してデータを確実に共有することができる。

【 0 1 0 7 】

また、O S 管理情報 O S 1 の変更内容やその変更履歴を画面に表示することにより、アプリケーション 6 0 や O S などの動作を視覚的に把握できデバッグ作業の効率を向上させることができる。

【 0 1 0 8 】

以上、本発明を具体的に説明したが、本発明は上記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で適宜変更可能であることはもちろんである。

【 0 1 0 9 】

本発明に係るデバッグ支援装置が携帯電話などの移動体端末に適用された例を示したがこれに限定されるものではない。例えば、ネットワーク端末等、マルチタスク処理が要求されるリアルタイム O S が組み込まれた装置で実行されるアプリケーションのデバッグ作業に適用できる。

【 0 1 1 0 】

また、O S 管理情報の種別ごとにそれぞれ共有ファイルのメモリ領域を決めておけば、イベントフラグの値やタスクの状態だけでなく、セマフォの状態、メールボックスの状態などの他の O S 管理情報も同様な手順で変更履歴を表示させることができる。

【 0 1 1 1 】

また、O S シミュレータによって制御される O S 管理情報を共有ファイル上に記憶し、記憶手段と共有ファイルとを共通化するようにしてもよい。具体的には

、OSシミュレータが共有ファイル上のOS管理情報の更新をすることにより、アプリケーションとOSデバッガとでOS管理情報の共有することができるようになる。これにより、アプリケーションが停止した場合でもOSデバッガによってOS管理情報が参照することができるようになる。

【0112】

さらに、本発明のプログラムをCD-ROM、FD、DVD等の記録媒体に格納し、汎用コンピュータに対して着脱自在な形で提供される形態でもよい。これにより、本発明のプログラムが格納されたソフトウェア商品として装置と独立して容易に配布、販売することができるようになる。汎用コンピュータや汎用ゲーム装置などのハードウェアを用いてこのソフトウェアを使用することにより、これらのハードウェアで本発明が容易に実施できるようになる。

【0113】

加えて、本発明を実現するためのプログラムやデータは、通信回線などを介して接続された他の機器から受信してメモリに記録する形態であってもよい。さらには、通信回線などを介して接続された他の機器側のメモリに上記プログラムやデータを記録し、このプログラムやデータを通信回線などを介して使用する形態であってもよい。

【0114】

【発明の効果】

本発明によれば、OS管理情報の変更と同期して、OS管理情報の変更が共有ファイルの共有管理情報に反映されるので、アプリケーションが停止した場合でもOS管理情報を容易に参照することができる。これにより、効率の良いデバッグを実行することができるようになる。

【0115】

また、OS管理情報をOSデバッガから変更できるため、OS管理情報が不正な値となった場合でも、アプリケーションの動作検証を継続することや、アプリケーションのプログラムを修正することなく割り込みが発生した状況をつくることが可能である。

【0116】

更に、アプリケーションとOSデバッガとの間で双方向で通信が実行されるまでアプリケーションの動作が停止されるので、OSデバッガが、OS管理情報の変更を取りこぼしなく取得することができる。

【図面の簡単な説明】

【図 1】

本発明の第 1 の実施の形態におけるデバッグ支援装置の機能ブロック図

【図 2】

本発明の第 2 の実施の形態におけるデバッグ支援装置の機能ブロック図

【図 3】

本発明の第 3 の実施の形態におけるデバッグ支援装置の機能ブロック図

【図 4】

同実施の形態に係るデバッグ支援装置の各手段の処理の流れを示すフローチャート

【図 5】

同実施の形態にかかる複数のタスクの状態遷移が表示された画面の一例を示す図

【図 6】

従来におけるデバッグ支援システムの概略機能ブロック図

【図 7】

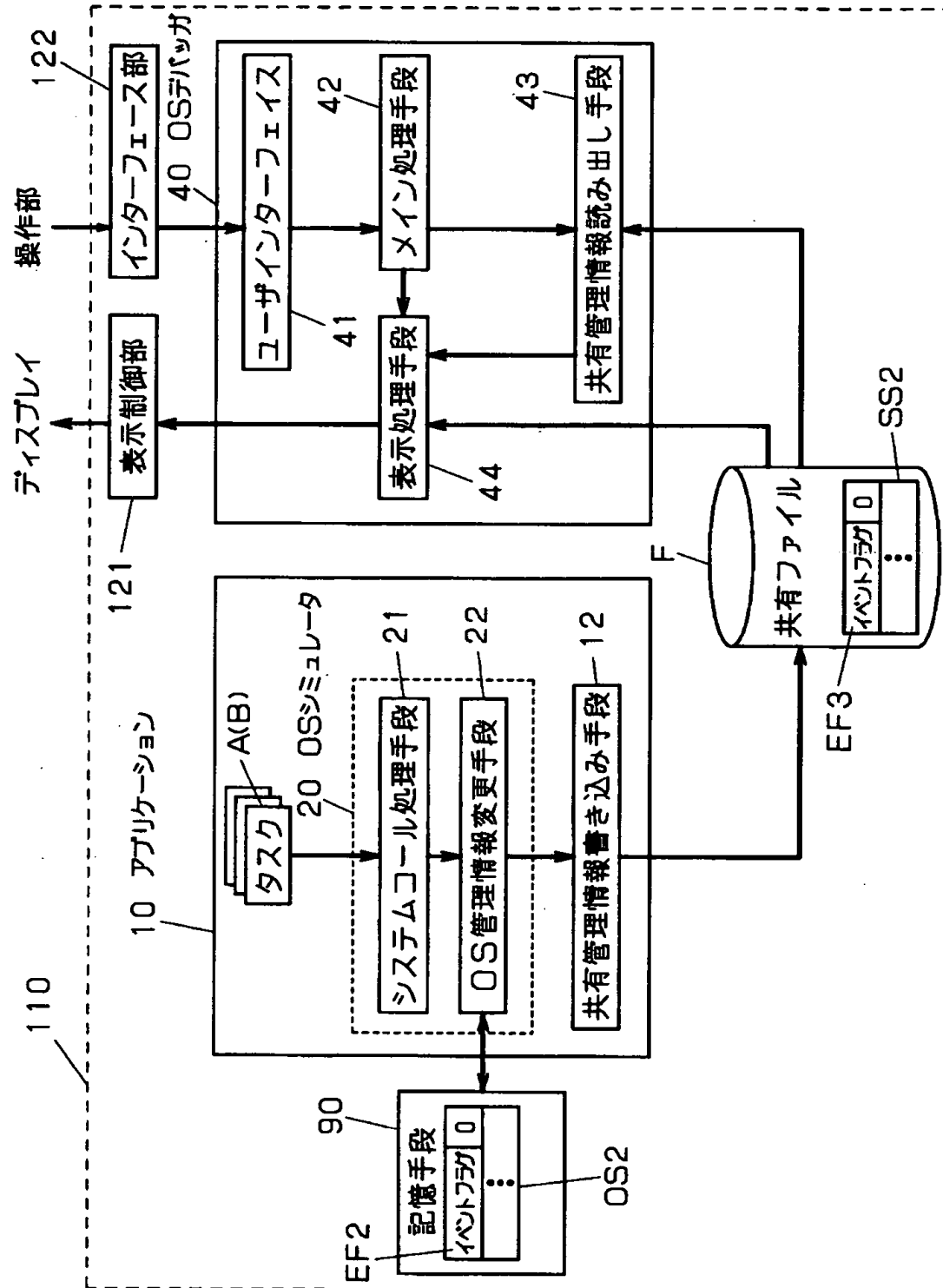
組み込み型アプリケーションの開発環境の説明図

【符号の説明】

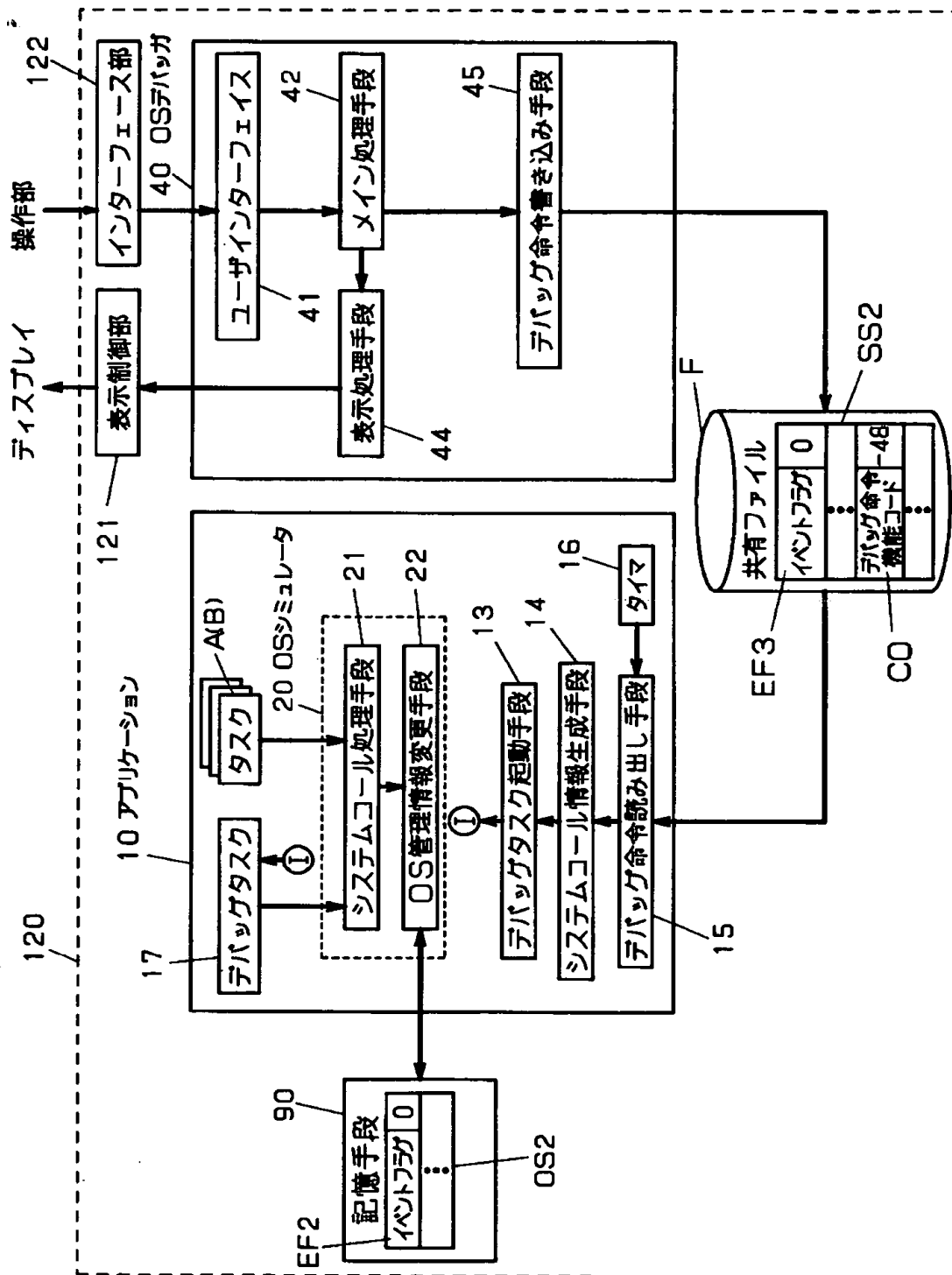
- 1 0 アプリケーション
- 1 2 OS管理情報書き込み手段
- 1 3 デバッグタスク起動手段
- 1 4 システムコール情報生成手段
- 1 5 デバッグ命令読み出し手段
- 1 7 デバッグタスク
- 2 0 OSシミュレータ
- 2 1 システムコール処理手段

- ・ 2 2 共有管理情報変更手段
- ・ 3 0 アプリケーションデバッガ
- ・ 4 0 O S デバッガ
- 4 1 ユーザインターフェイス
- 4 2 メイン処理手段
- 4 3 共有管理情報読み出し手段
- 4 4 表示処理手段
- 4 5 デバッグ命令書き込み手段
- A, B タスク

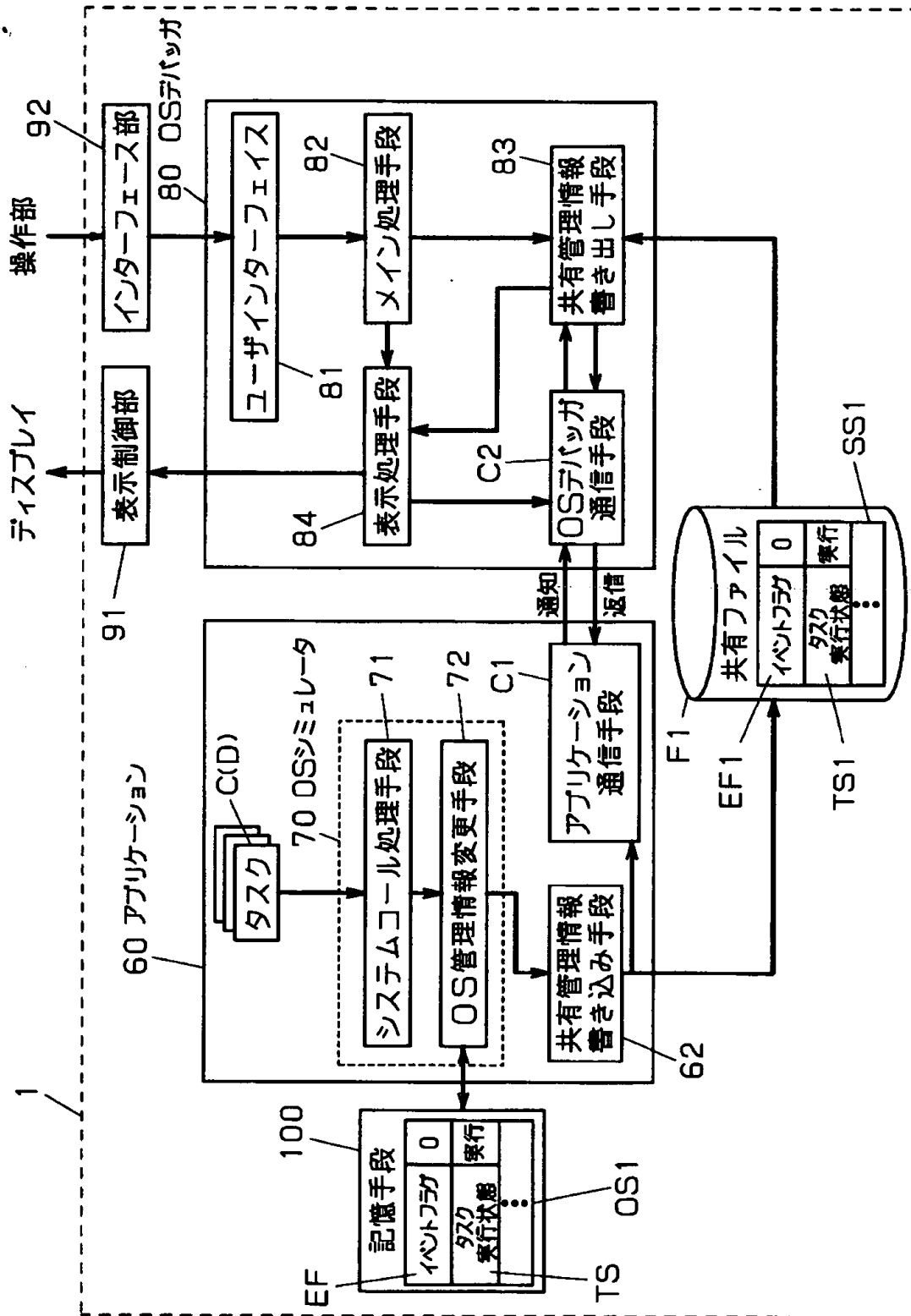
【書類名】 図面
【図 1】



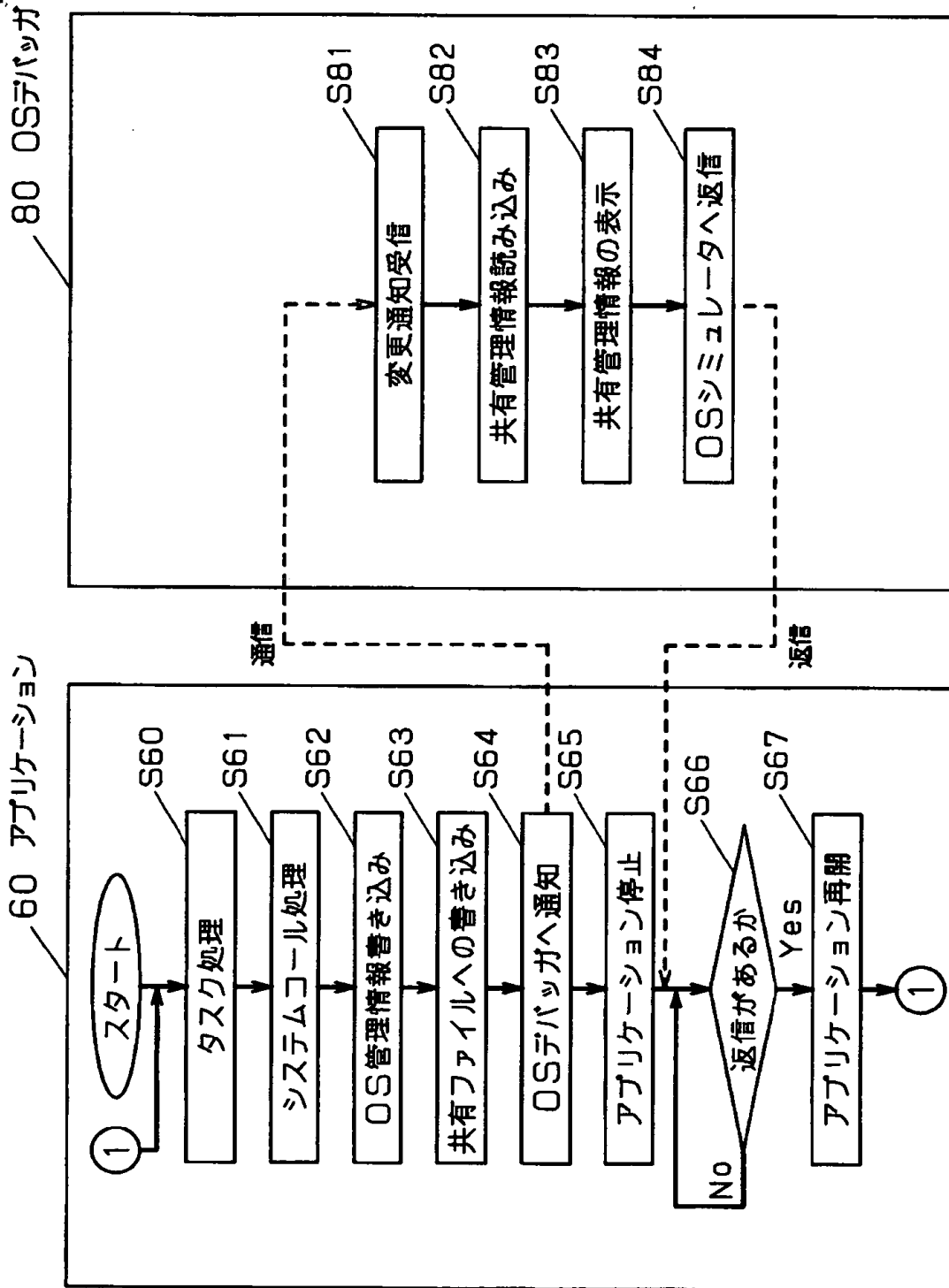
【図2】



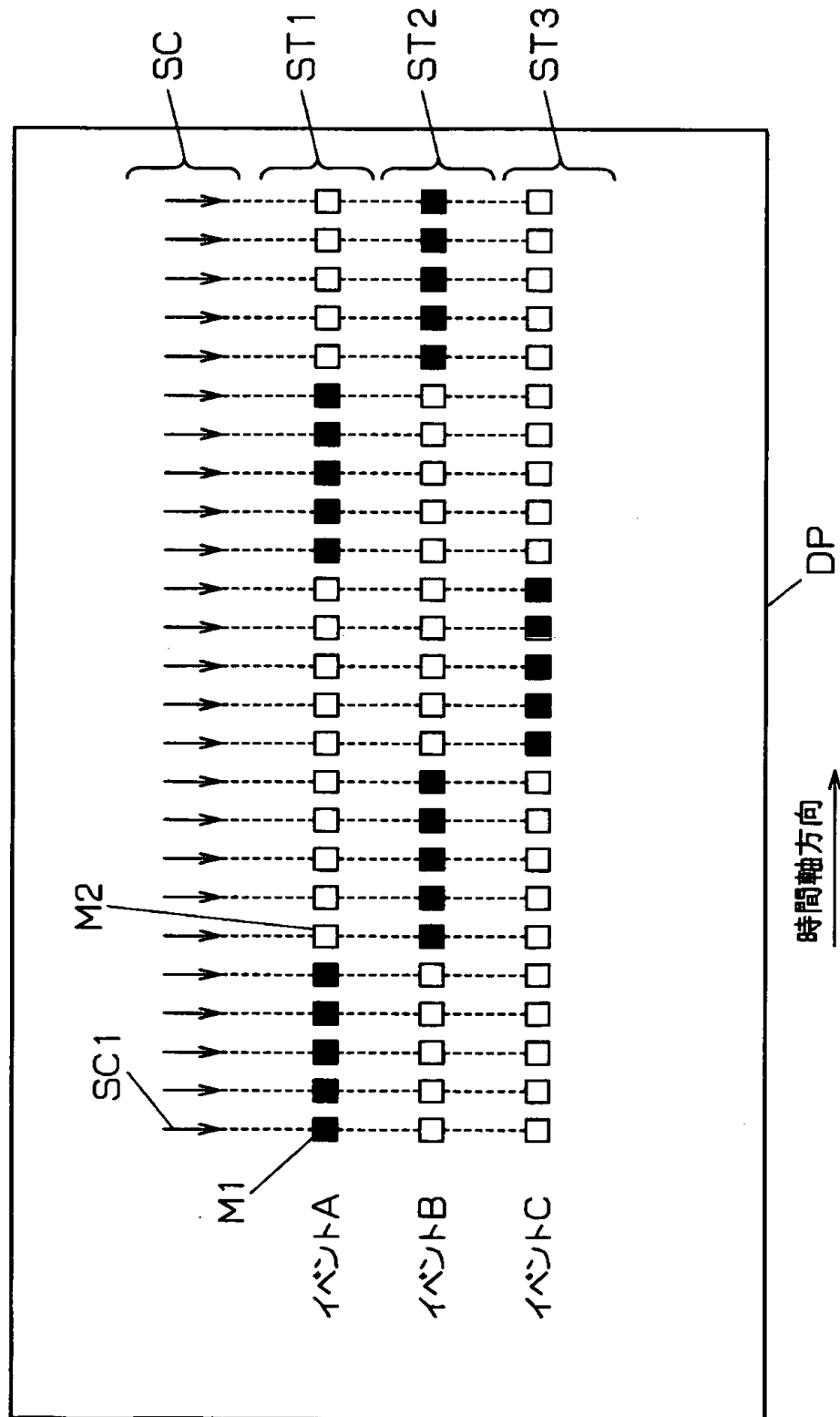
【図 3】



【図4】

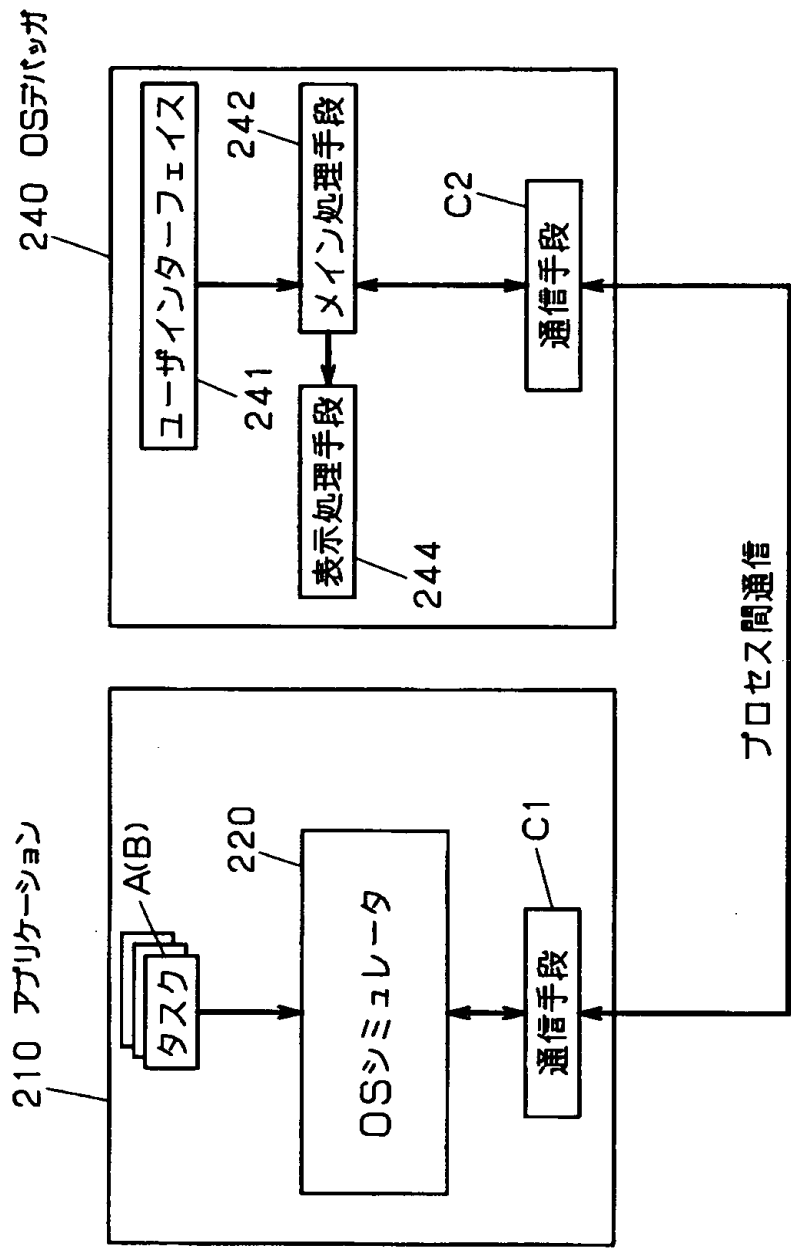


【図5】



従来におけるデバッグ支援システムの概略機能ブロック図

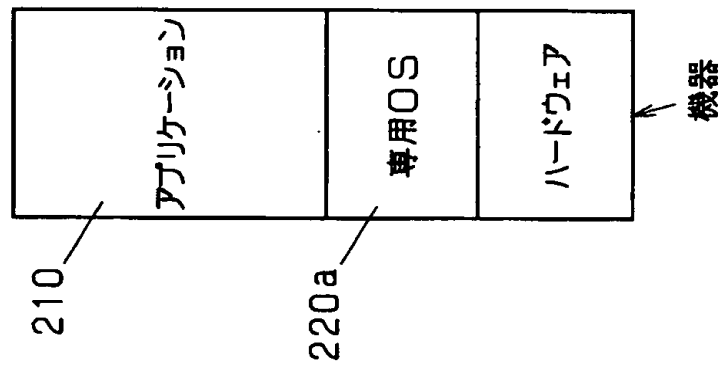
【図 6】



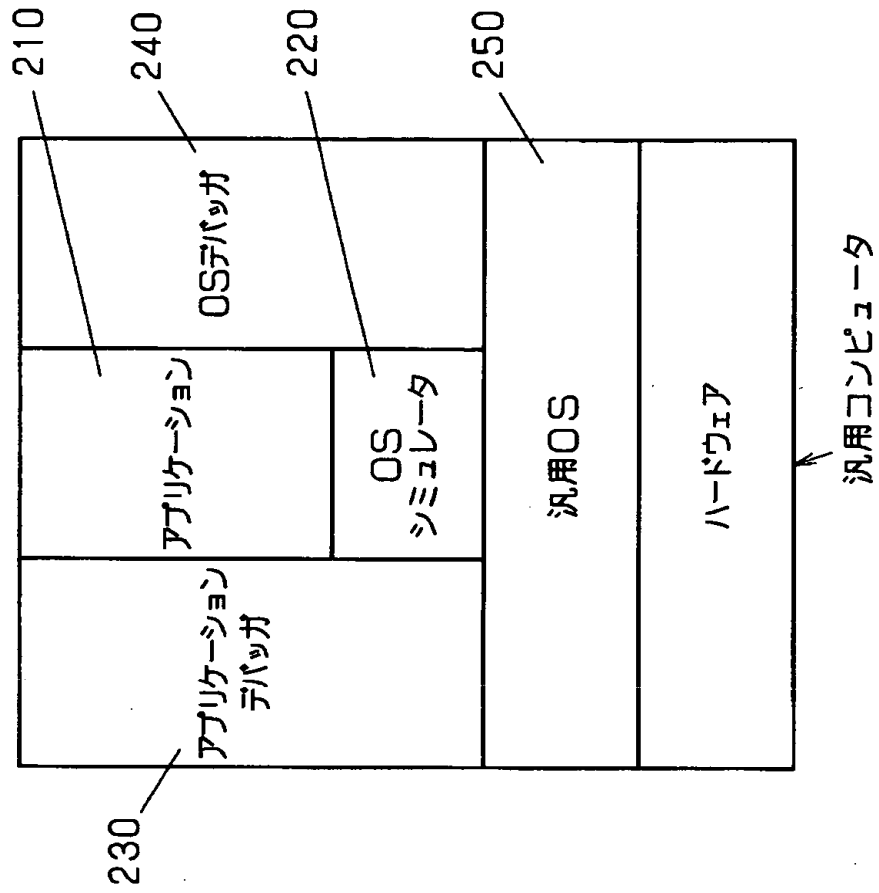
【図 7】

組み込み型アプリケーションの開発環境

(a)



(b)



【書類名】 要約書

【要約】

【課題】 OSシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のOS管理情報を参照できるようにしたデバッグ支援システムを提供する。

【解決手段】 OSシミュレータがリンクされたアプリケーションと、リアルタイムOSの動作を仮想的に制御するOSデバッガとを有するシミュレーション環境において、アプリケーションとOSデバッガとで共有される共有ファイルに、OS管理情報と同一のデータを含む共有管理情報が記憶される。共有ファイルの共有管理情報へのOS管理情報の書き込みや、書き込まれた共有管理情報を読み出すようにする。

【選択図】 図1

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 8 2 1]

1. 変更年月日 1 9 9 0 年 8 月 2 8 日

[変更理由] 新規登録

住 所 大阪府門真市大字門真 1 0 0 6 番地

氏 名 松下電器産業株式会社